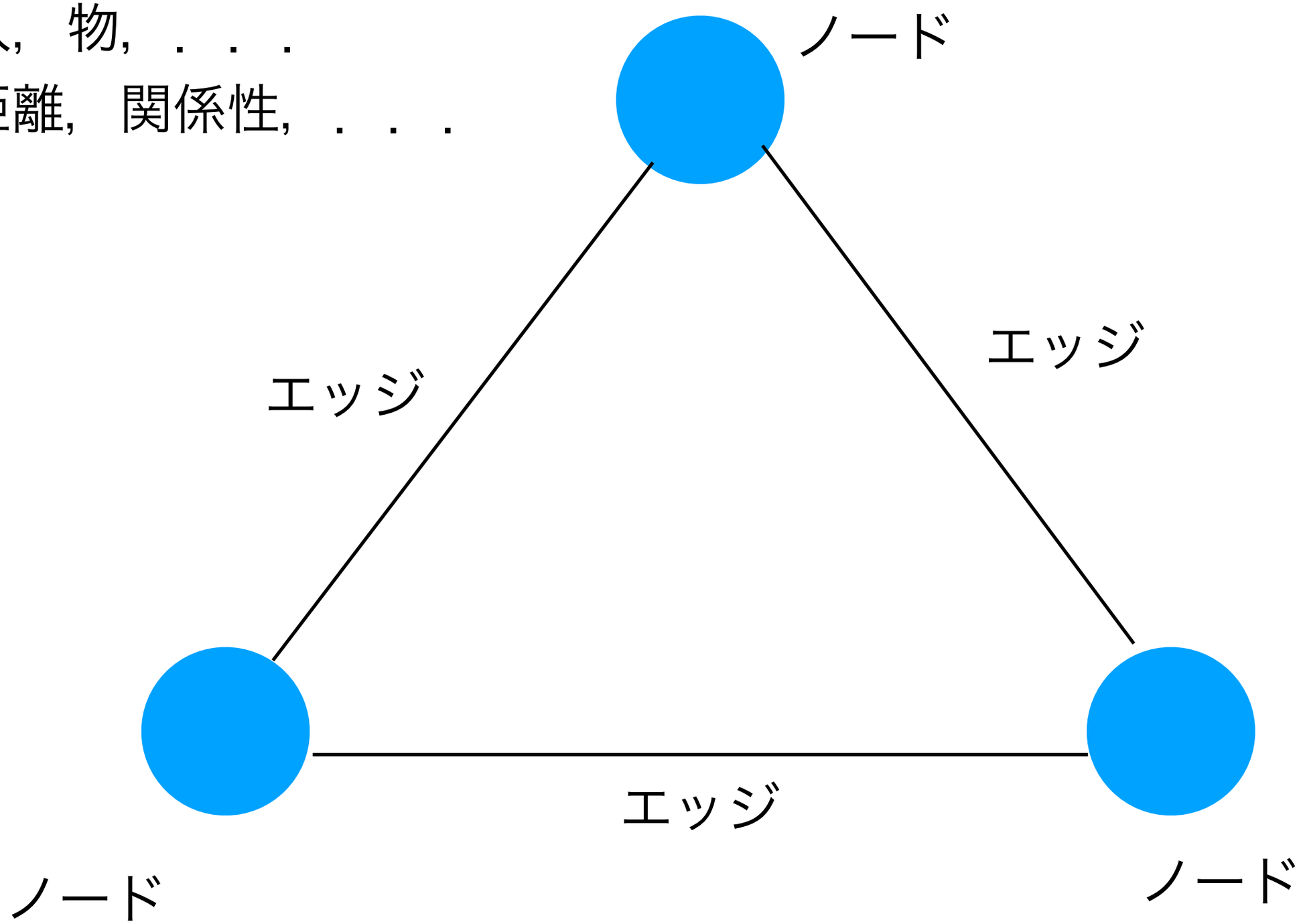
グラフの利用

NetworkXからGNNへ

グラフの概念

ノード：人，物，．．．

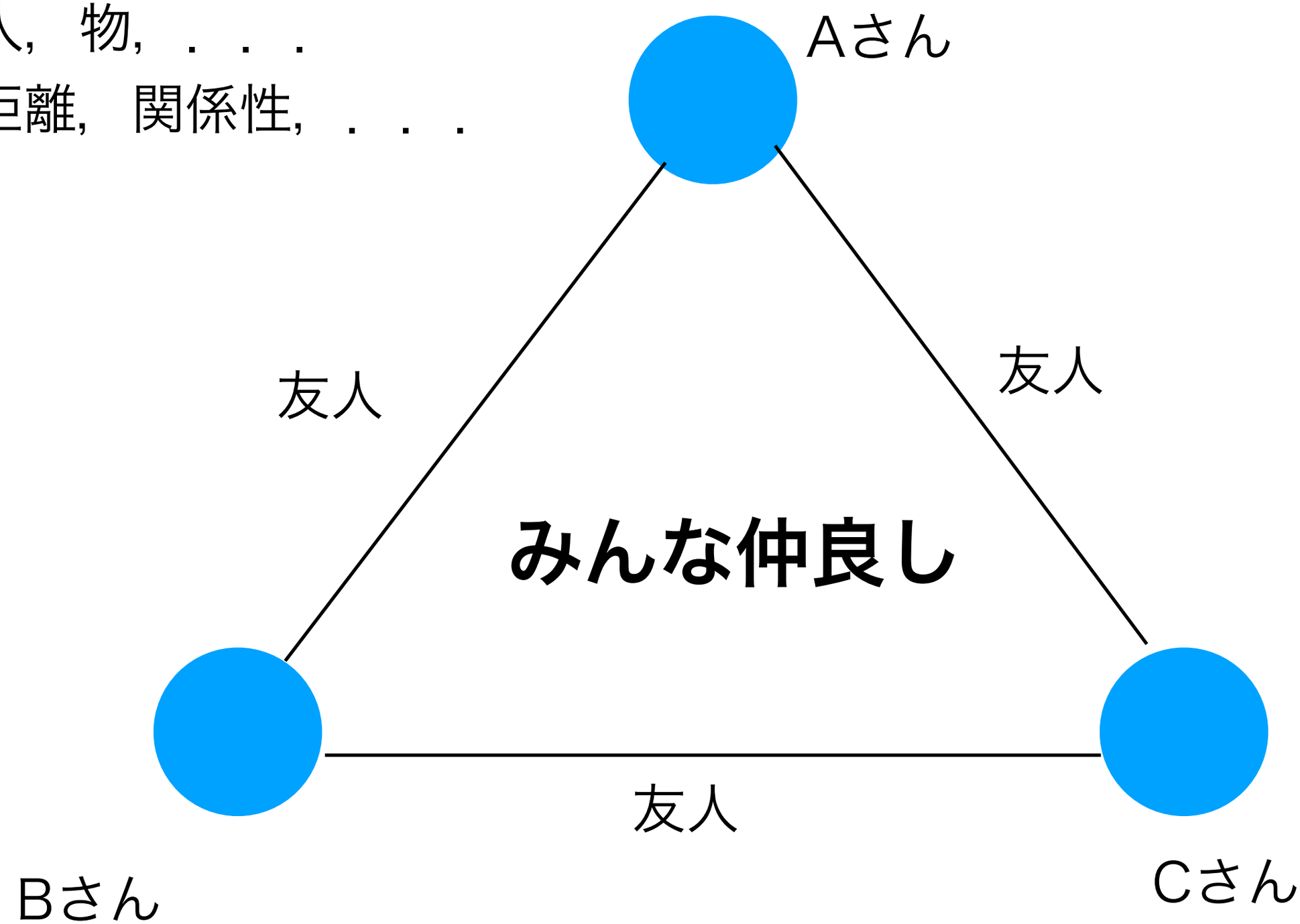
エッジ：距離，関係性，．．．



グラフの概念

ノード：人，物，．．．

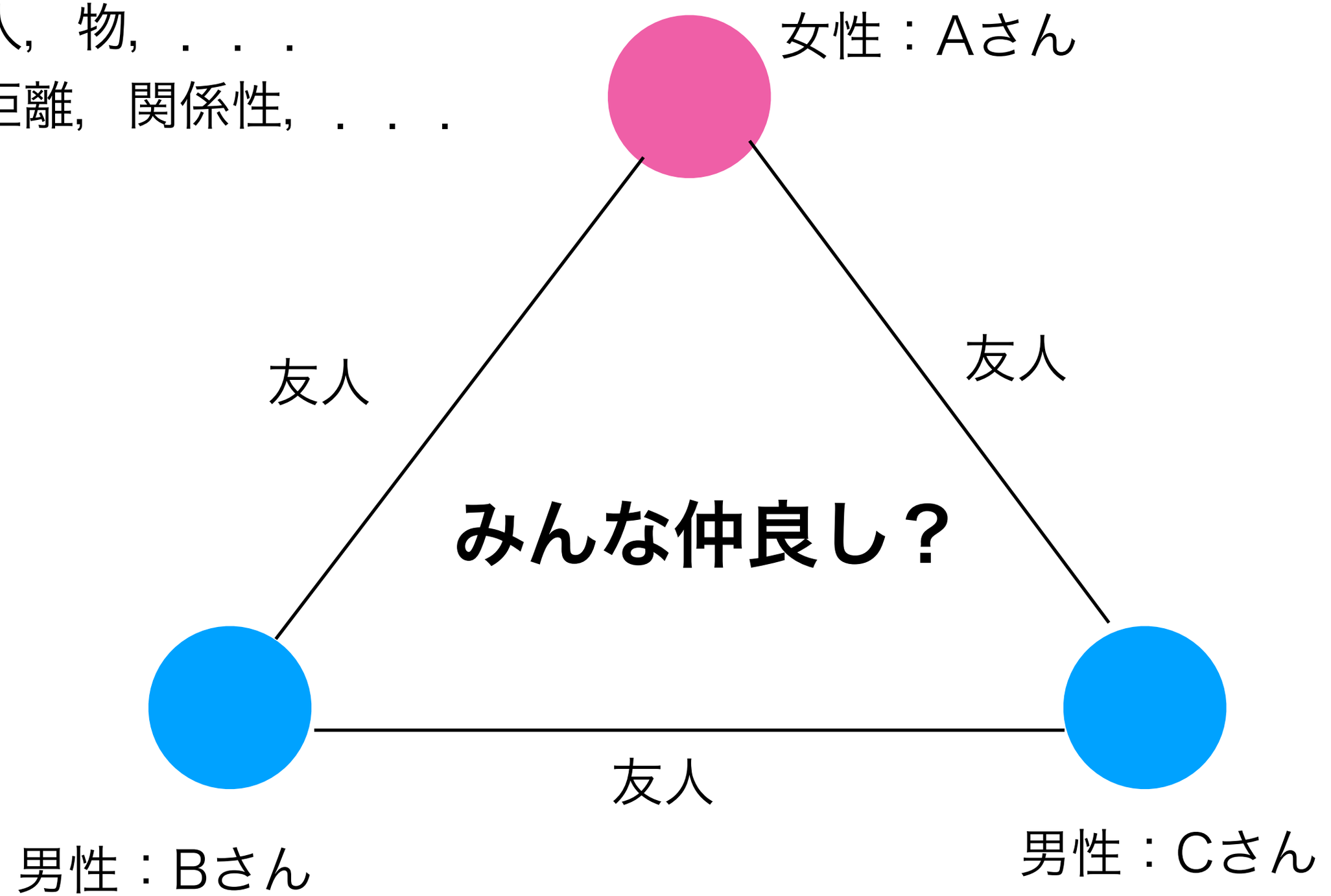
エッジ：距離，関係性，．．．



グラフの概念

ノード：人，物，．．．

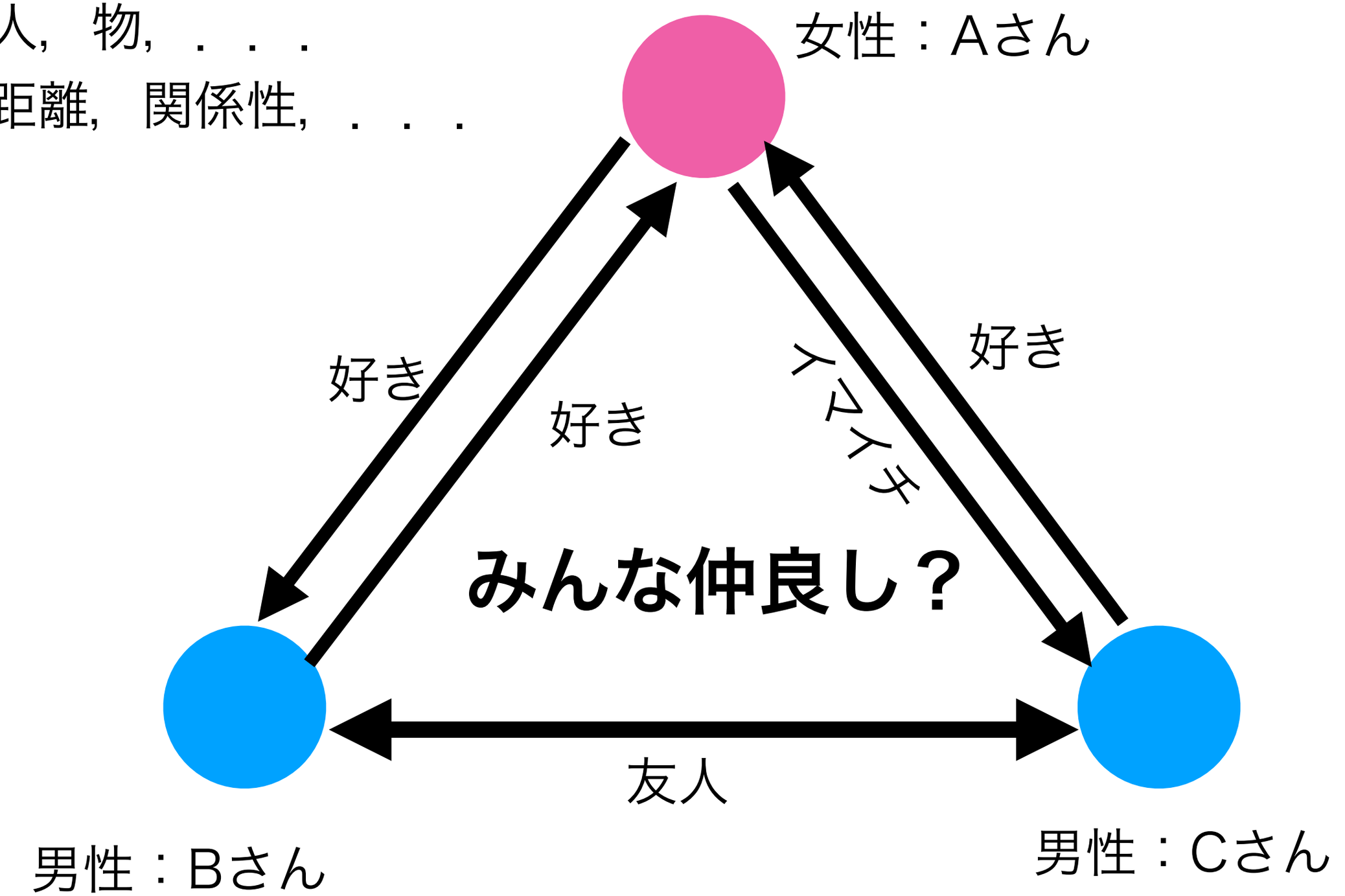
エッジ：距離，関係性，．．．



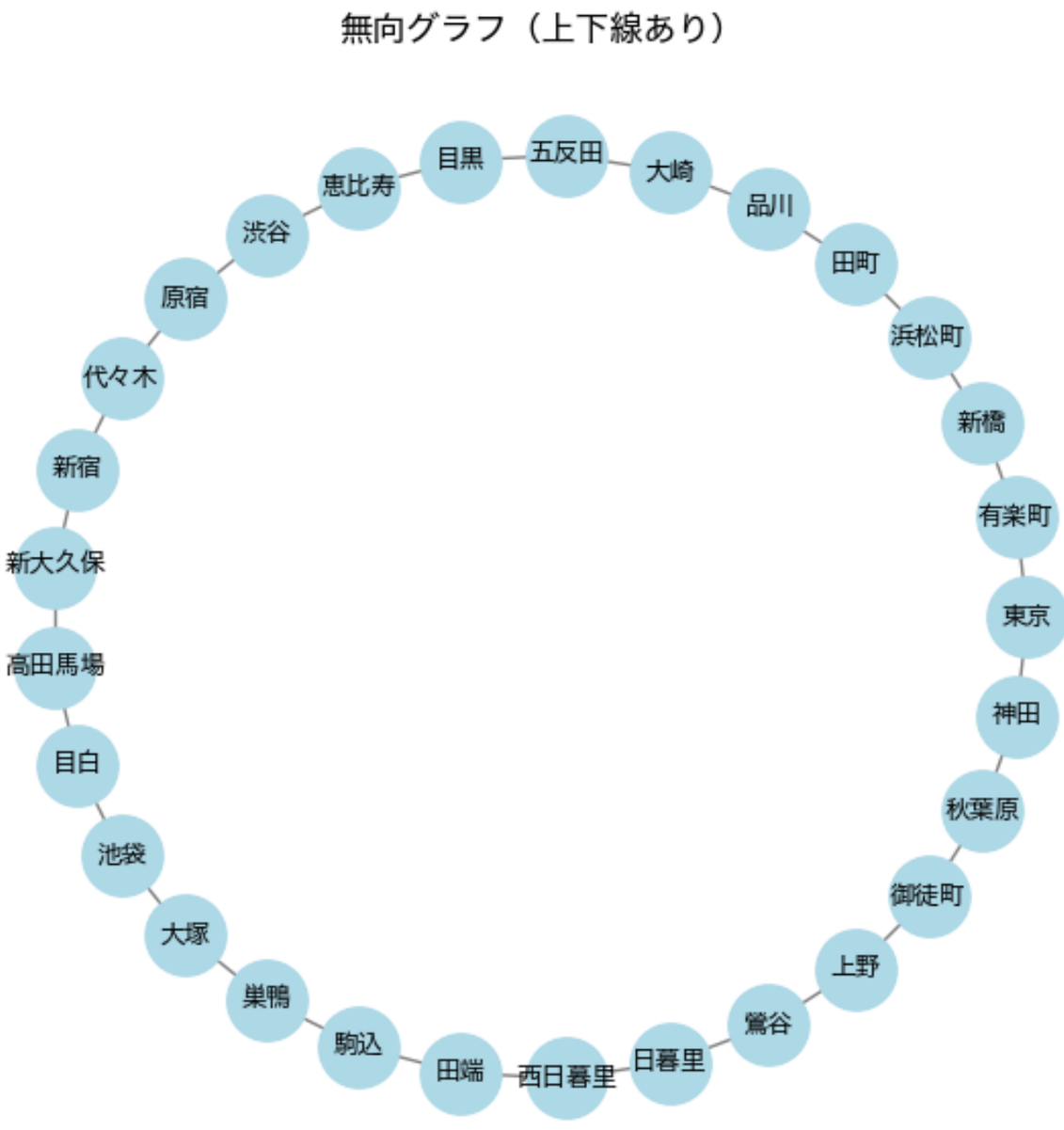
グラフの概念

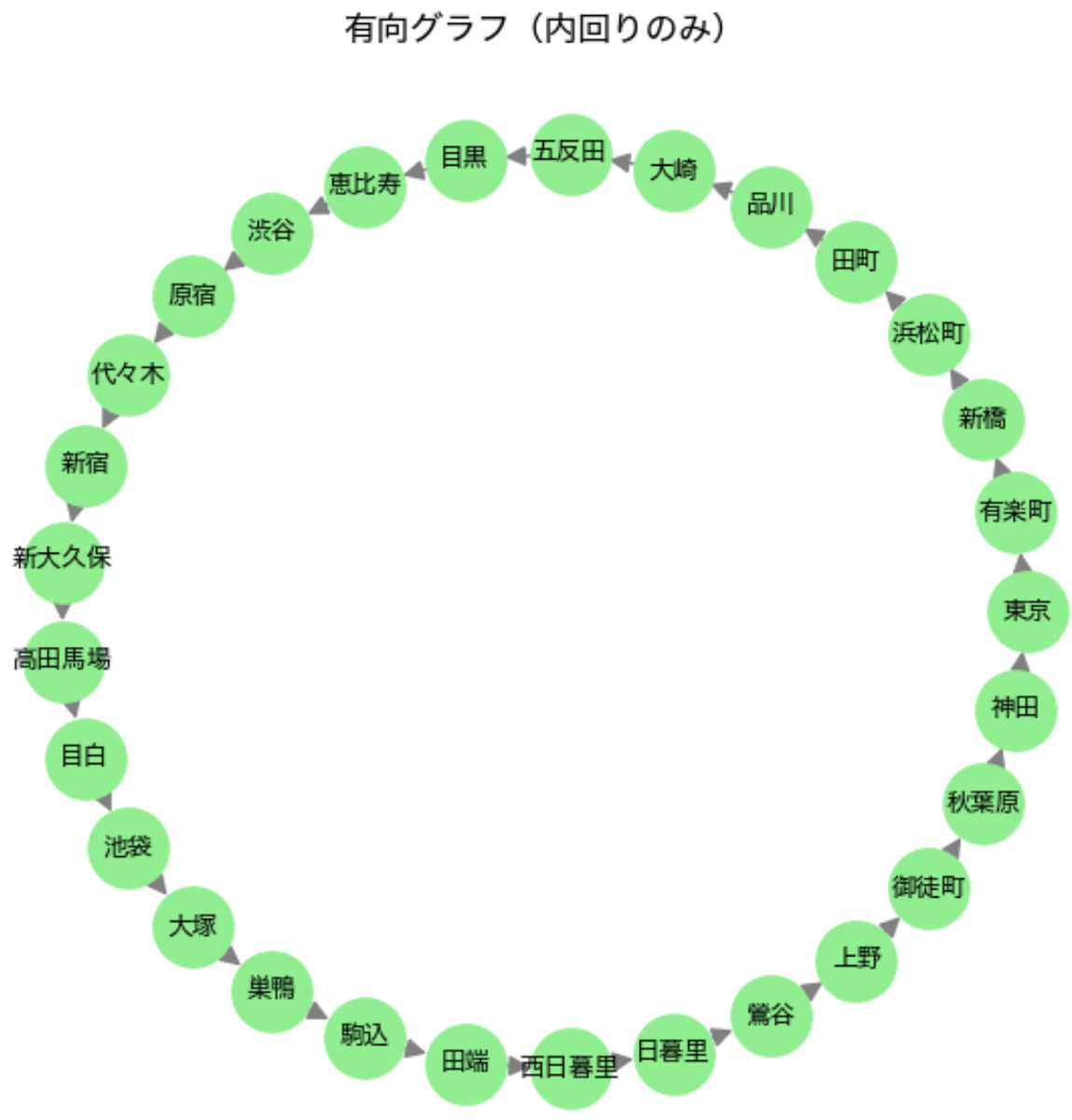
ノード：人，物，．．．

エッジ：距離，関係性，．．．



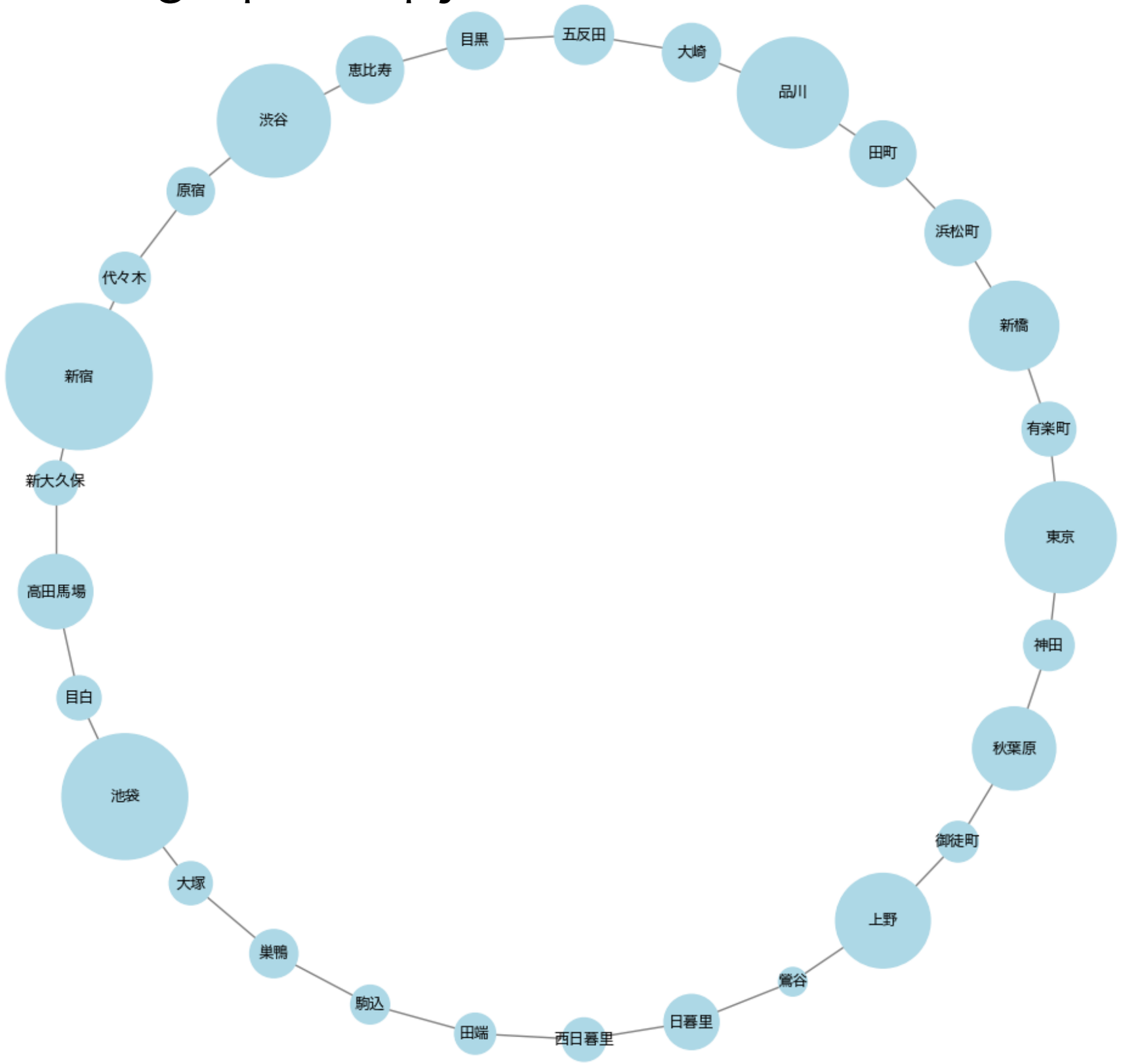
001_YamanoteLine_graph01.ipynb





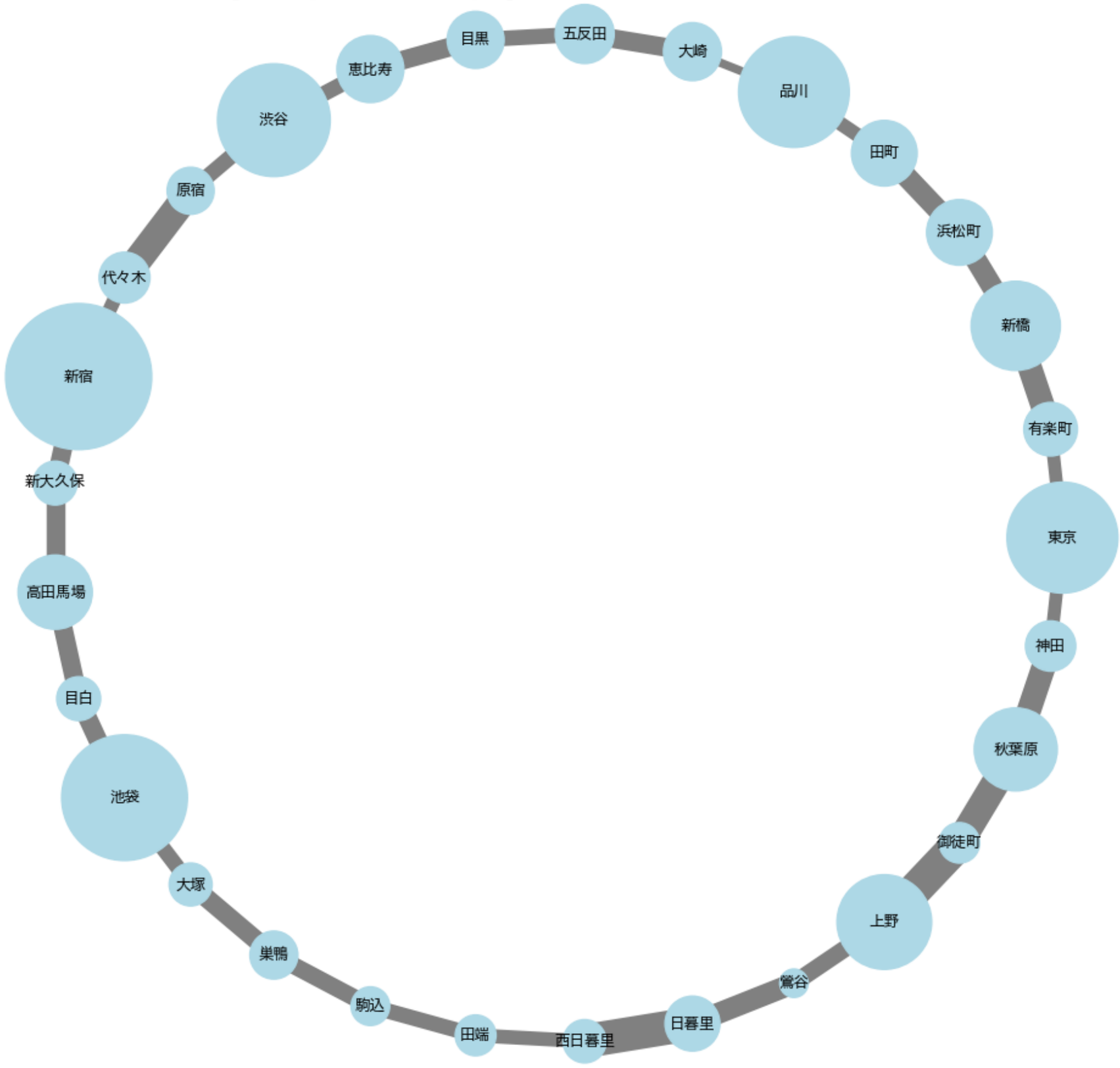
山手線無向グラフ：駅の利用者数に応じたノードサイズ

002_YamanoteLine_graph01.ipynb



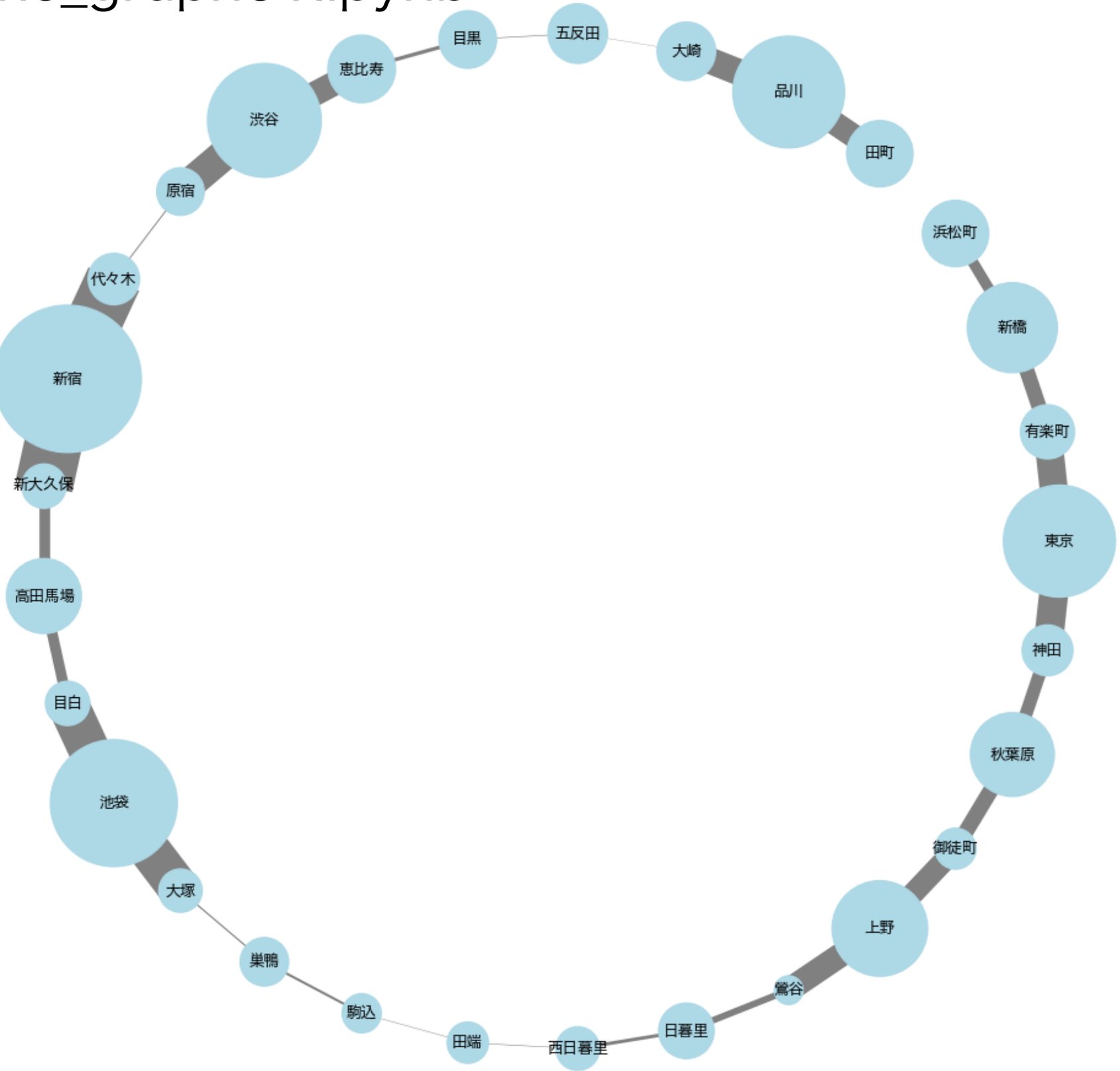
山手線無向グラフ：距離でエッジの太さ、乗車数でノードサイズ表示

003_YamanoteLine_graph01.ipynb

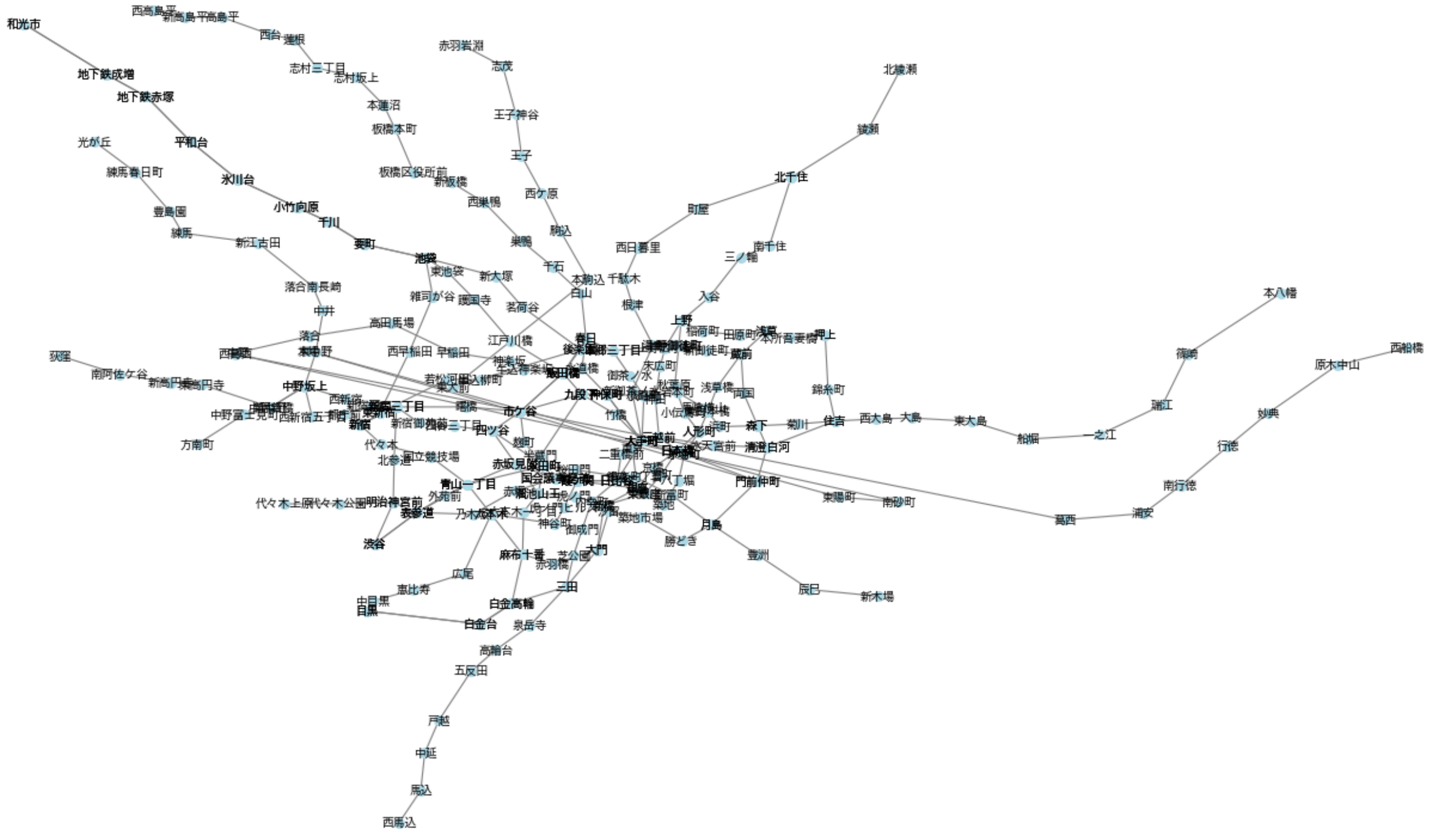


山手線：乗車人数の差でエッジの太さを表現（距離は考慮せず）

004_YamanoteLine_graph01.ipynb



005_TokyoMetro.ipynb



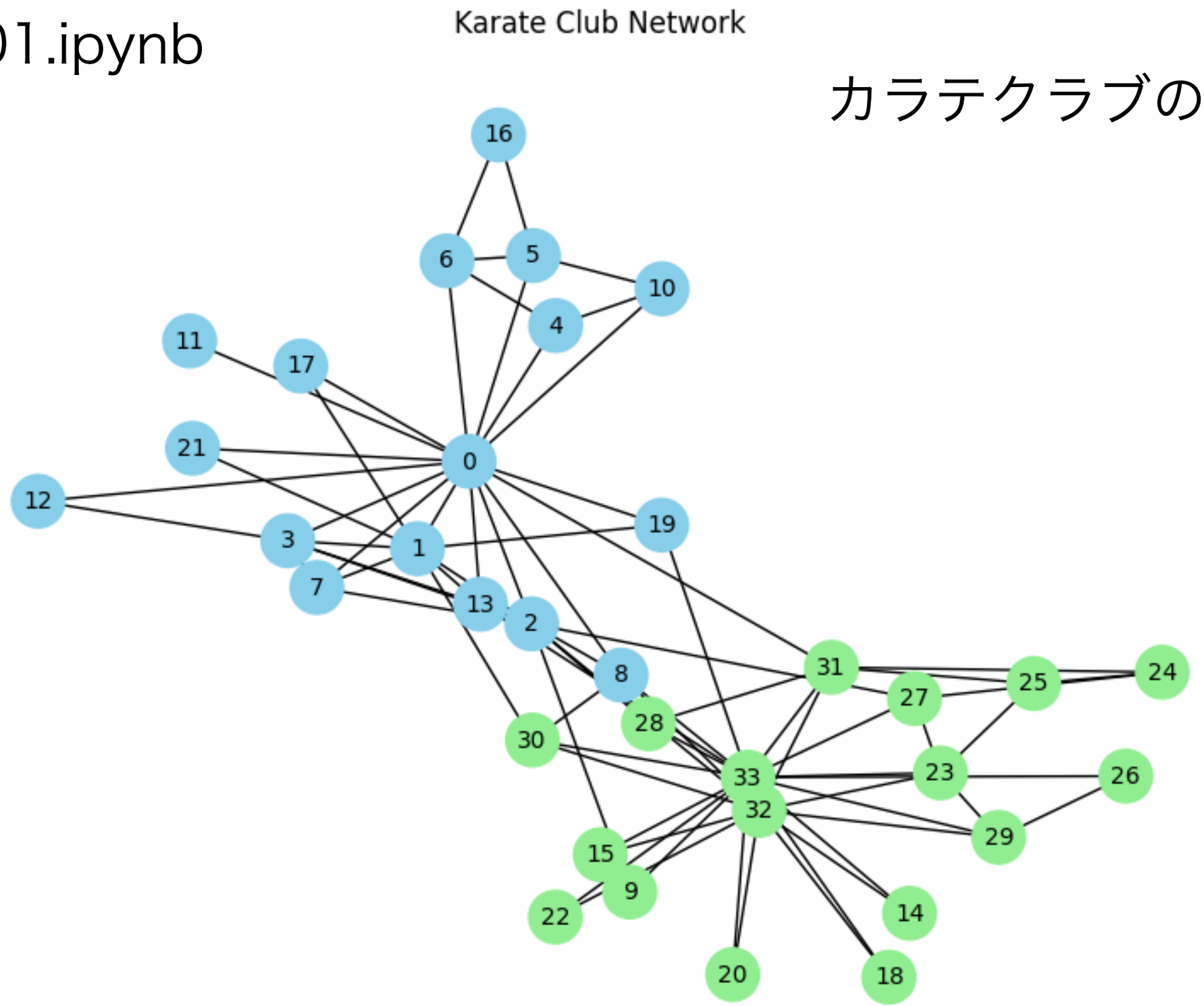
005_TokyoMetro.ipynb

乗換案内

[15]:
```
###  乗り換え案内

# 出発・到着の駅ID
start_id = "A01"  #  西馬込
goal_id = "Z14"   # 押上

# 最短経路（乗車距離に基づく）
shortest_path = nx.shortest_path(G, source=start_id, target=goal_id, weight='weight')

# 距離
total_distance = nx.shortest_path_length(G, source=start_id, target=goal_id, weight='weight')

# 駅ID→日本語名の辞書
id_to_jname = {station_id: info["name_jp"] for station_id, info in metro["stations"].items()}

# 経路を駅名で表示
station_names_path = [id_to_jname.get(sid, sid) for sid in shortest_path]

# 表示
print("最短経路（駅名）:")
print(" → ".join(station_names_path))
print(f"\n総乗車距離: {total_distance:.2f} km")
```

```
最短経路（駅名）:
西馬込 → 馬込 → 中延 → 戸越 → 五反田 → 高輪台 → 泉岳寺 → 三田 → 大門 → 新橋 → 新橋 → 銀座 → 京橋 → 日本橋 → 日本橋 → 人形町 → 東日本橋 → 浅草橋 → 蔵前 → 浅草 → 本所吾妻橋 → 押上 → 押上

総乗車距離: 18.25 km
```

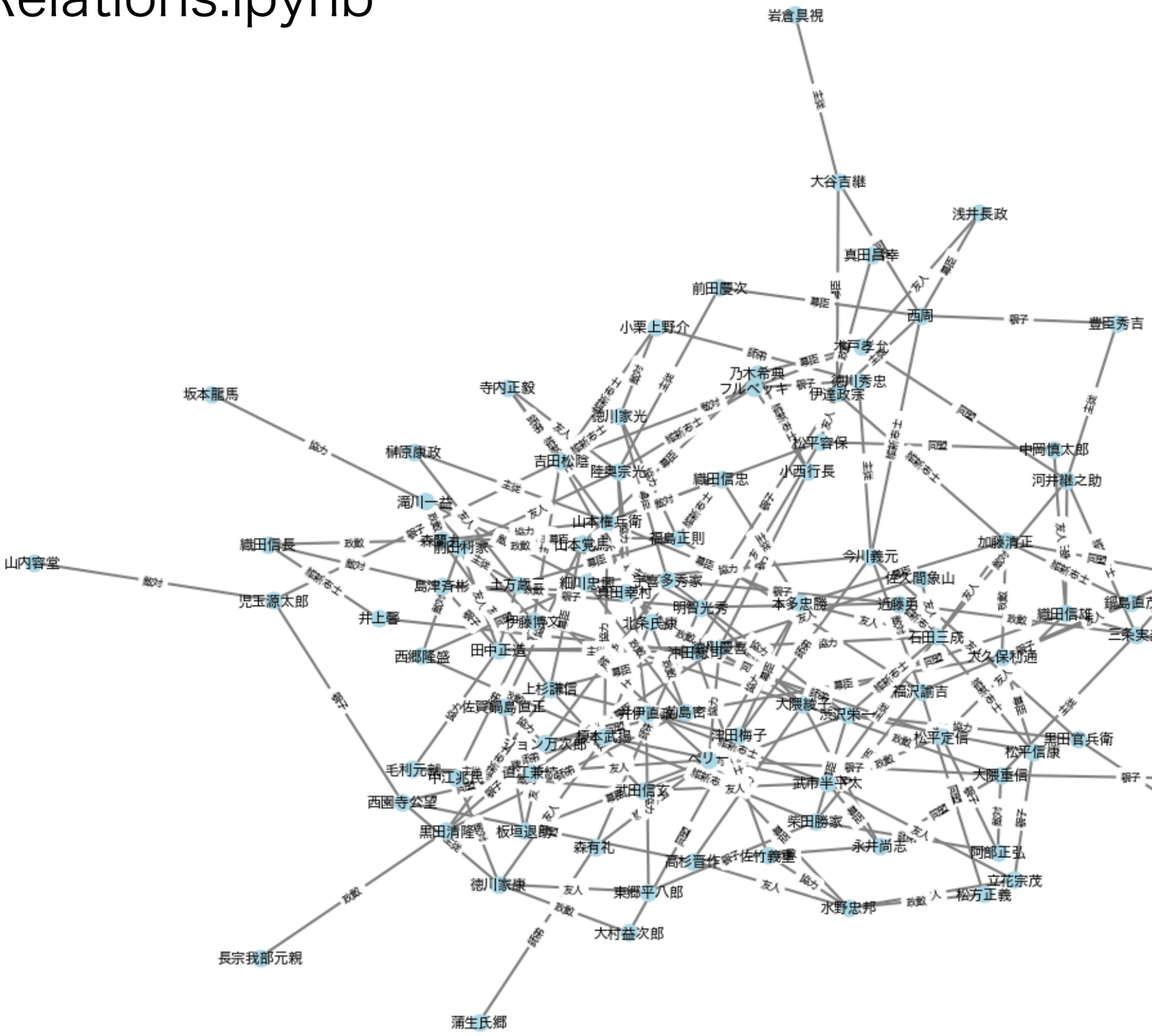
Karate01.ipynb



カラテクラブの分断

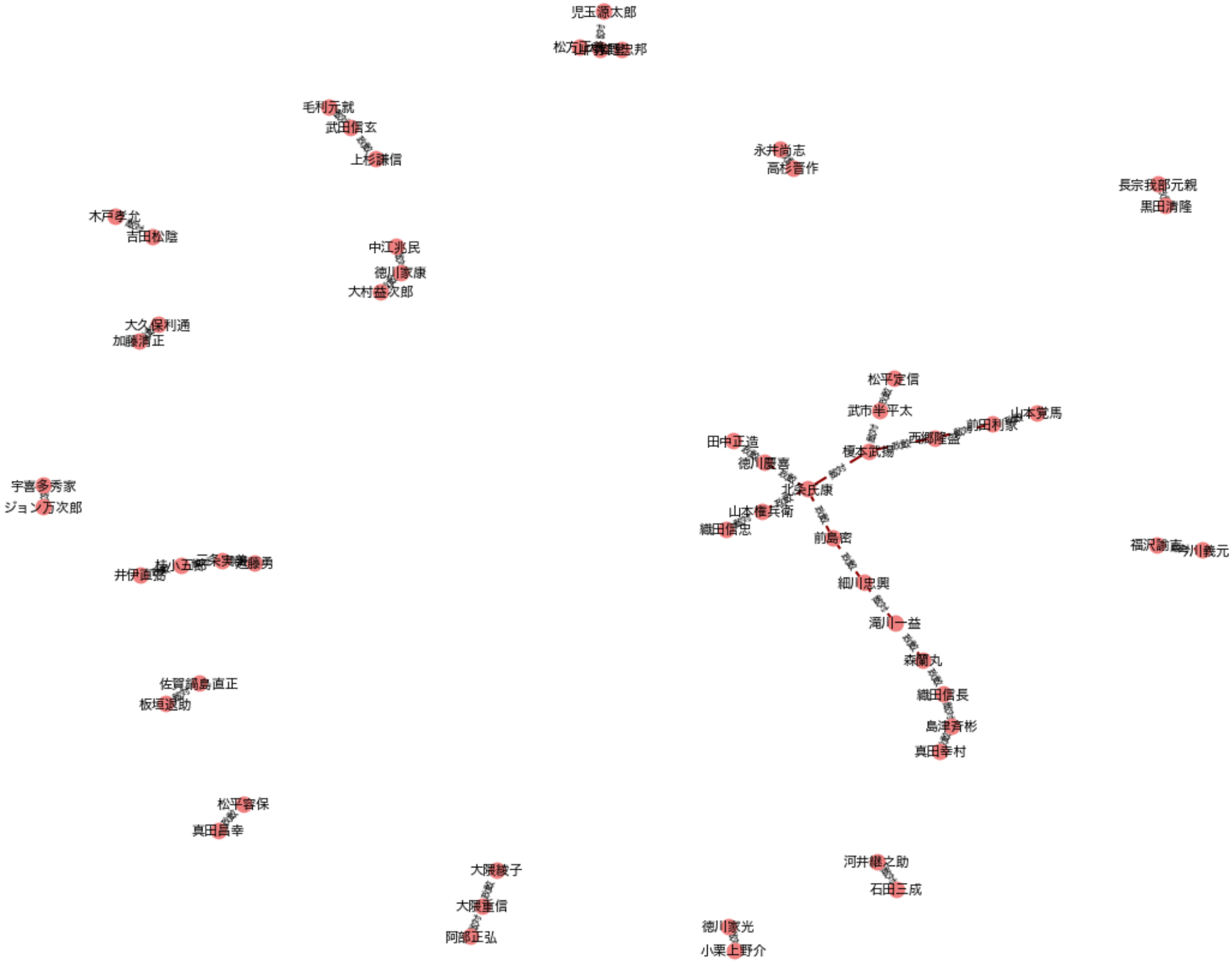
戦国〜明治 人物関係ネットワーク

001_Relations.ipynb



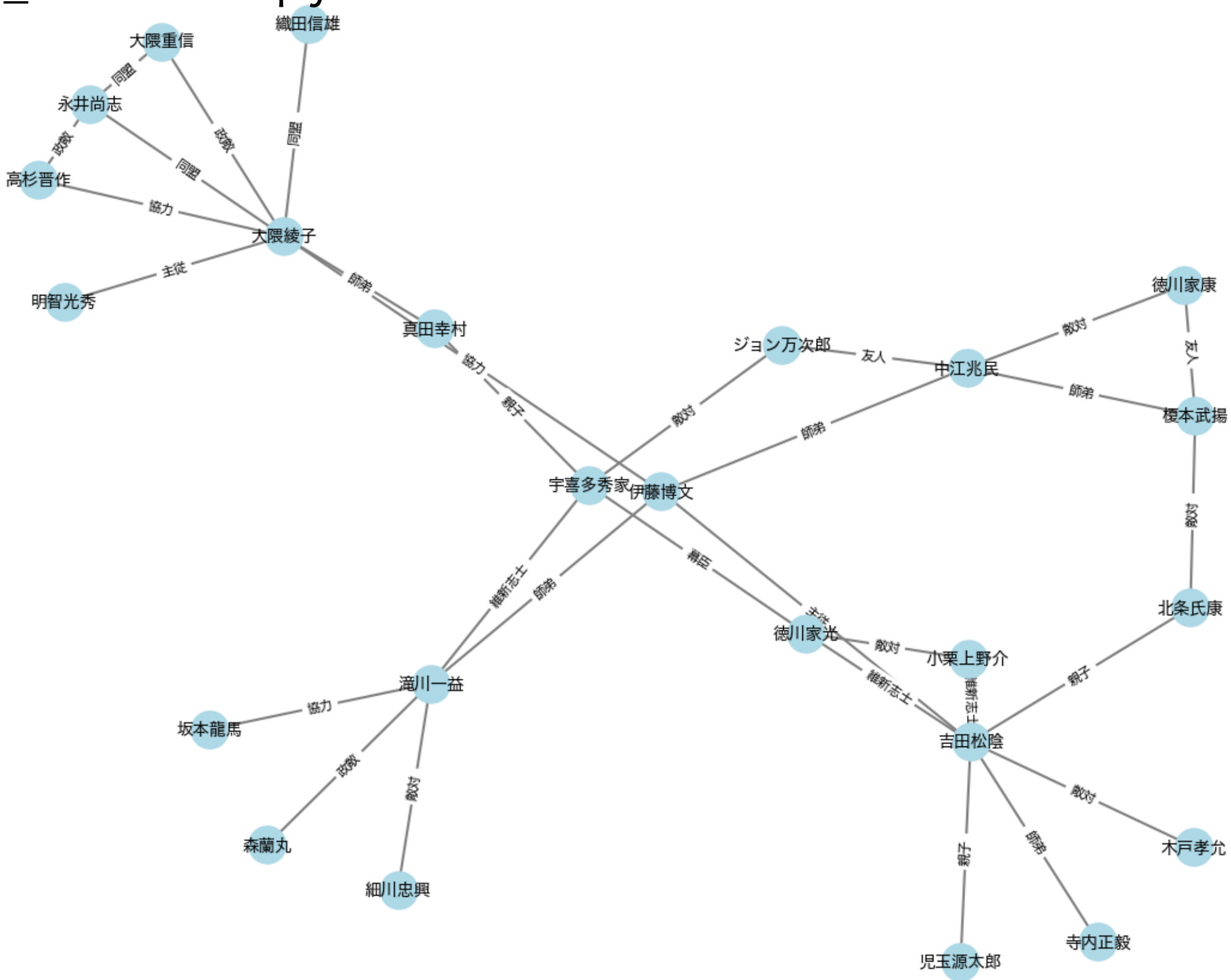
敵対関係のみの人物ネットワーク

001_Relations.ipynb



伊藤博文 と2ホップ以内の人物ネットワーク

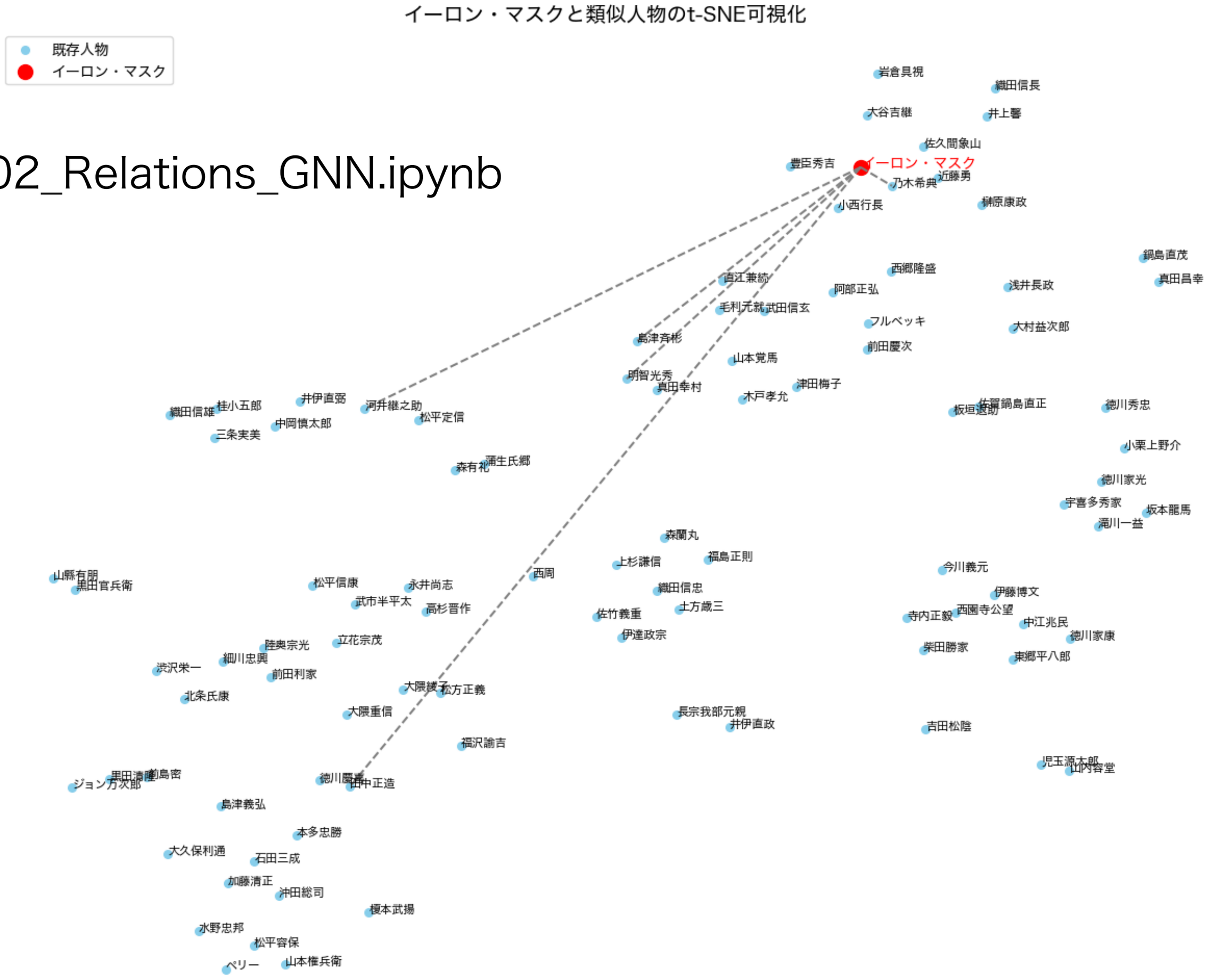
001_Relations.ipynb



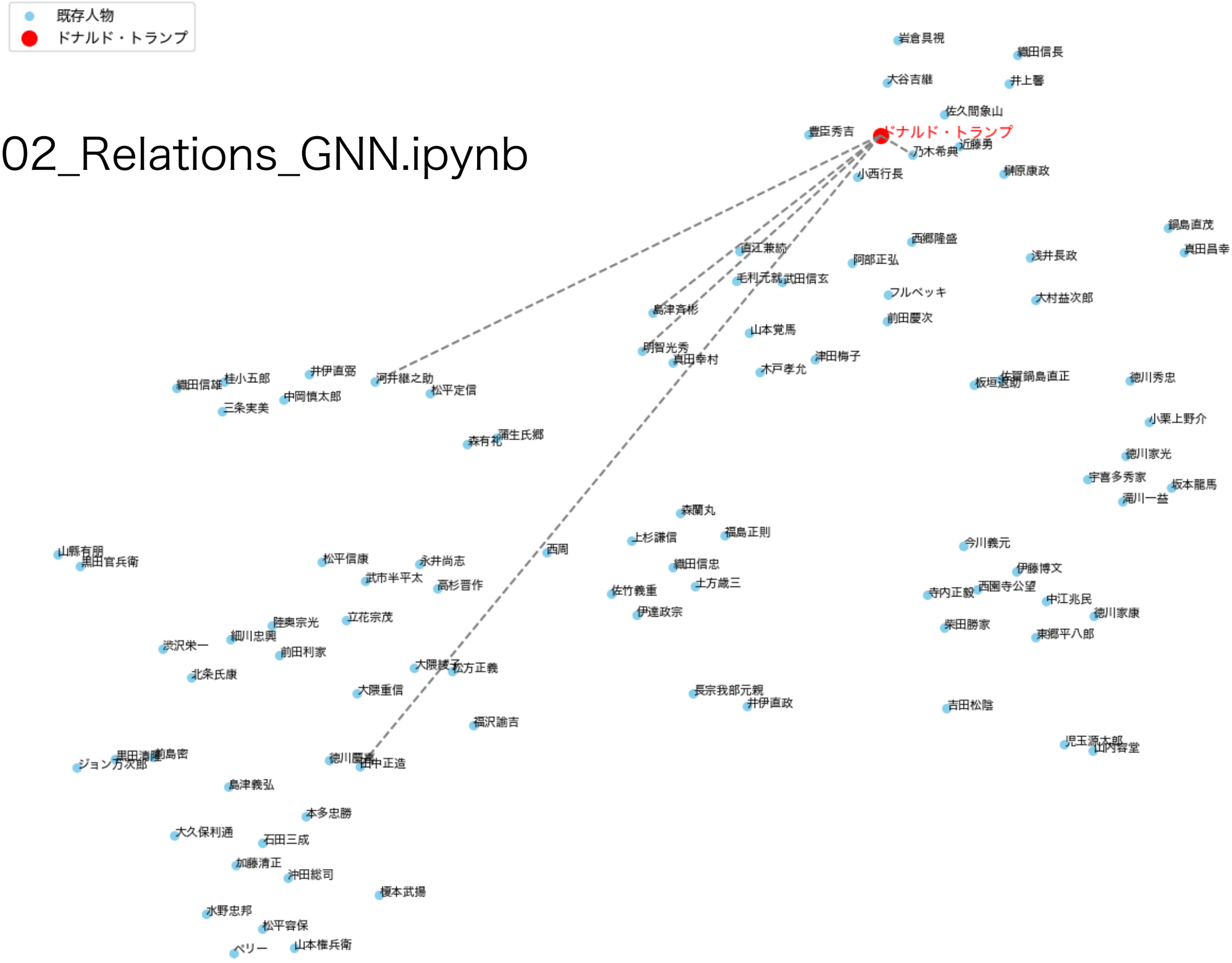
t-SNEによる人物グラフの埋め込み可視化

002_Relations_GNN.ipynb

002_Relations_GNN.ipynb

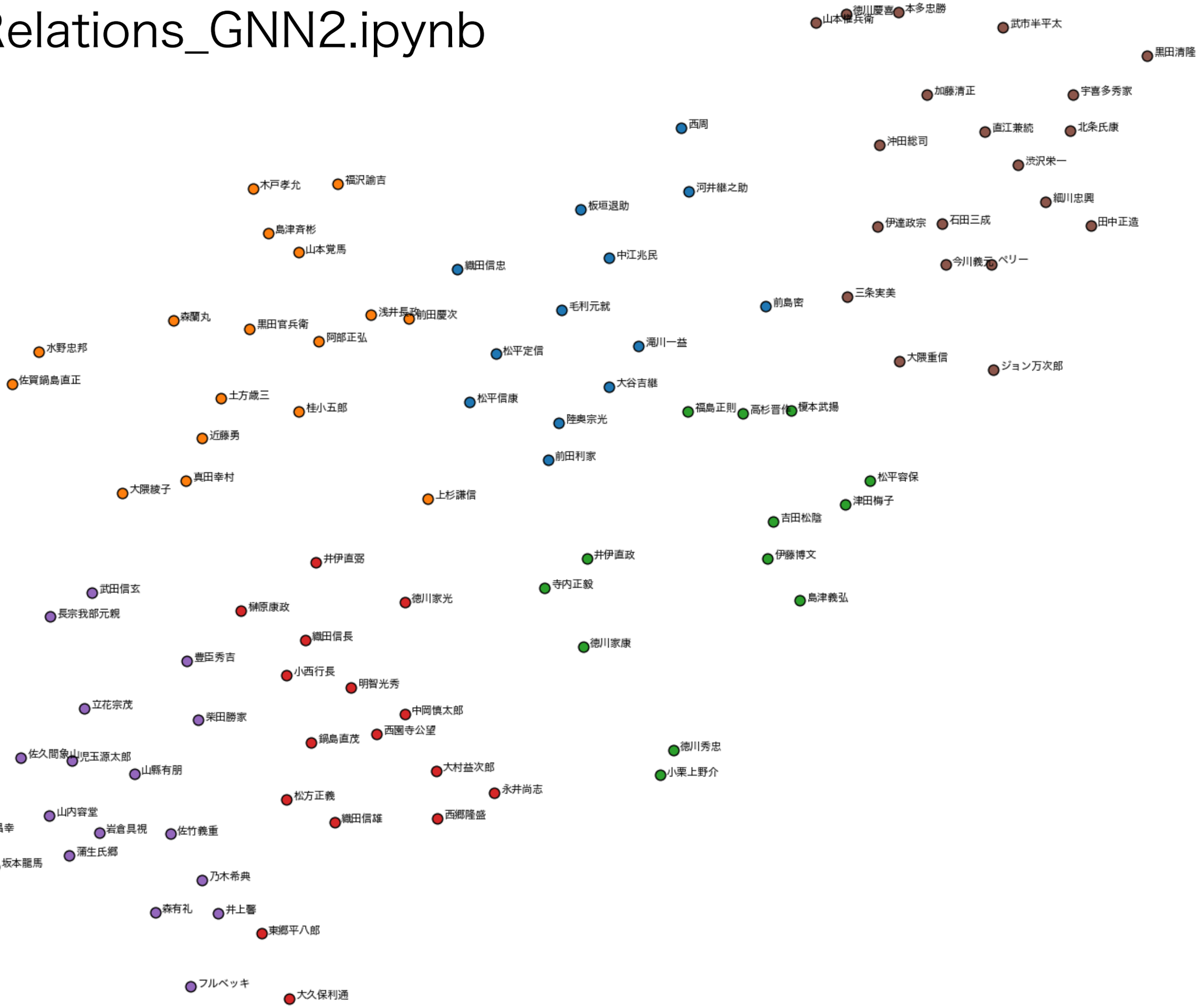


002_Relations_GNN.ipynb



R-GCN による人物の潜在空間クラスタリング（t-SNE + KMeans）

003_Relations_GNN2.ipynb



003_Relations_GNN2.ipynb

```python
#イーロン・マスクが戦国時代にいたら、誰と同盟を結びそうか？
#関係を問わず似ているベクトルとの比較

#イーロンマスクの性格を推定
elon_vec = (z[id1] + z[id2]) / 2  # 例：織田信長と豊臣秀吉の中間

# 内積スコアで全ノードとの類似度を計算
scores = torch.matmul(z, elon_vec)  # shape: (num_nodes,)

# 上位N人を表示
top_n = 10
top_indices = scores.topk(top_n).indices.tolist()

print("イーロン・マスクが同盟を結びそうな人物:")
for idx in top_indices:
    print(f"{id_to_name[idx]}: スコア = {scores[idx].item():.4f}")
```

```
イーロン・マスクが同盟を結びそうな人物:
山本権兵衛: スコア = 3162.9001
伊達政宗: スコア = 2756.8740
三条実美: スコア = 2704.5034
本多忠勝: スコア = 2601.1997
徳川慶喜: スコア = 2478.6990
西周: スコア = 2453.5803
武市半平太: スコア = 2317.7993
ジョン万次郎: スコア = 2275.4517
ペリー: スコア = 2266.1355
加藤清正: スコア = 2242.8247
```

```
イーロン・マスクと対立しそうな人物:
 1. 伊達政宗: スコア = 2756.8740
 2. 本多忠勝: スコア = 2601.1997
 3. 西周: スコア = 2453.5803
 4. ペリー: スコア = 2266.1355
 5. 沖田総司: スコア = 1976.5996
 6. 渋沢栄一: スコア = 1893.8912
 7. 直江兼続: スコア = 1852.9302
 8. 津田梅子: スコア = 1578.0886
 9. 井伊直政: スコア = 1571.2019
10. 前田慶次: スコア = 1329.1342
```

```
elon_vec
```

```
tensor([ -3.2888,  13.0683,   2.0526,  -5.1892,  -8.1813, -14.6359,  -1.4288,
         -6.7928,  17.0231, -10.7947,   1.8045,   0.0446,   4.5918,  -1.4434,
          7.7484,  -4.7431], device='mps:0', grad_fn=<DivBackward0>)
```

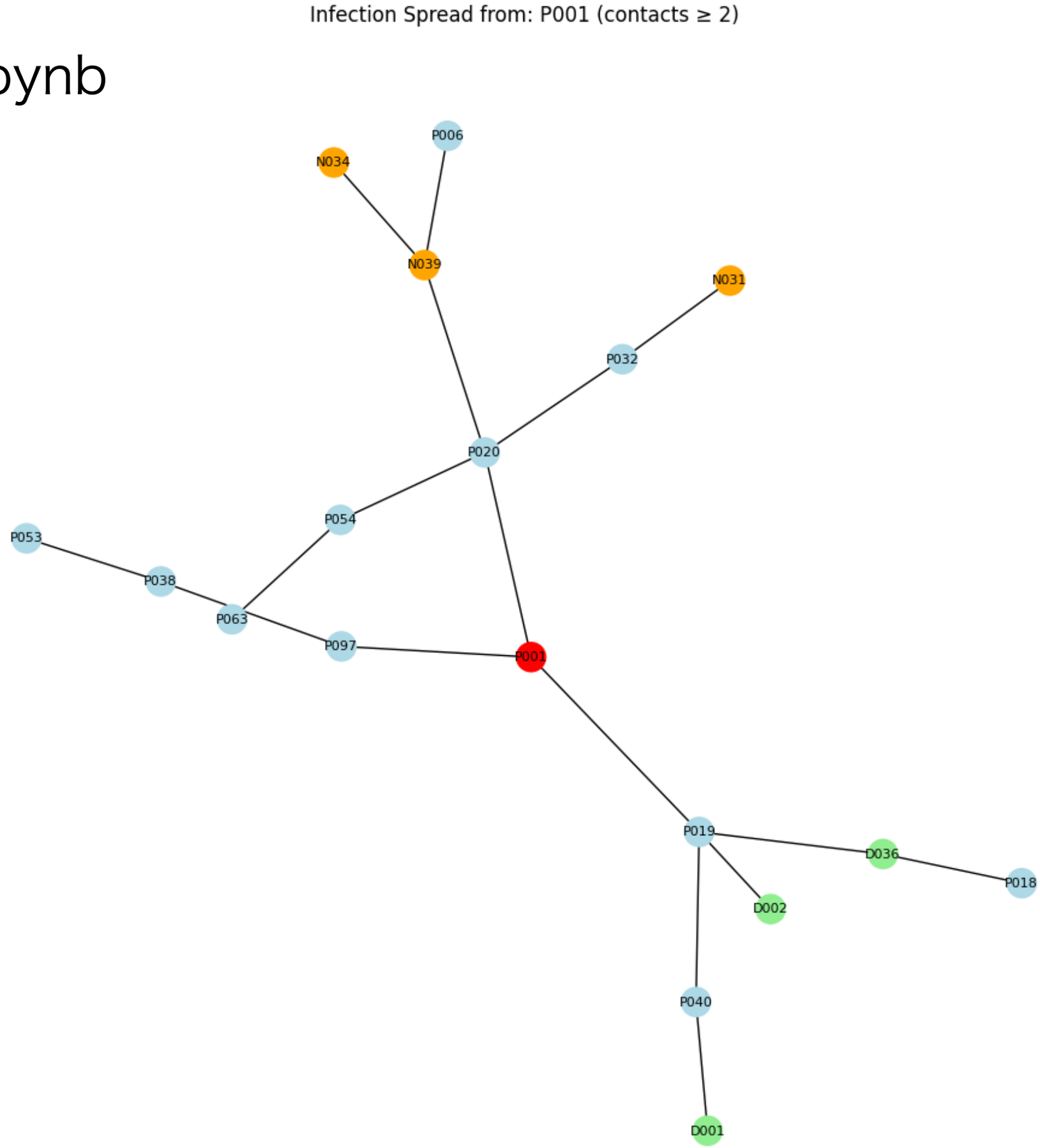
```
z[id1]
```

```
tensor([ -2.5097,  10.0491,   4.8908,  -7.7335,  -6.0620, -10.2900,  -2.5249,
         -5.5800,  15.5686, -12.8835,   4.2258,  -0.0943,   5.8782,   4.4728,
         12.4196,   0.6472], device='mps:0', grad_fn=<SelectBackward0>)
```

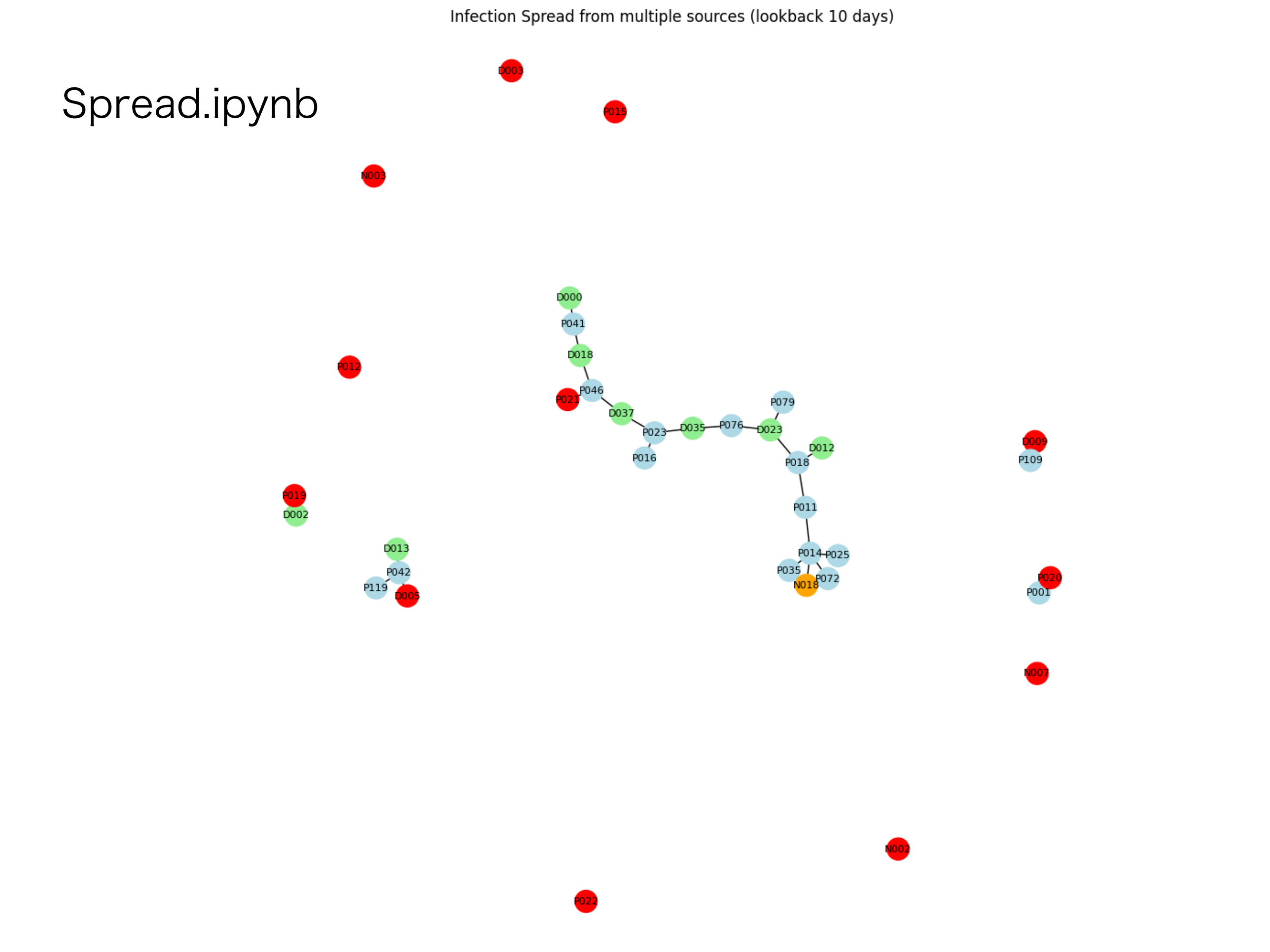
```
z[id2]
```

```
tensor([ -4.0679,  16.0876,  -0.7855,  -2.6450, -10.3006, -18.9818,  -0.3328,
         -8.0056,  18.4776,  -8.7060,  -0.6167,   0.1835,   3.3055,  -7.3595,
          3.0772, -10.1334], device='mps:0', grad_fn=<SelectBackward0>)
```

Spread.ipynb



Spread.ipynb



簡単なまとめ

グラフは抽象的な表現

グラフニューラルネットワークはグラフのノードを潜在ベクトル化

その解析は計算可能

意味は人が考える

計算可能な形で現実世界の出来事を計算可能

ぜひ演習にチャレンジしてください